1 QuickSort

We consider a variation of the QuickSort, which uses the following partition algorithm. Note that the rest of the QuickSort algorithm remains intact: will recursively sort the $A^-$ and $A^+$ partitions. For analysis, we assume the elements in $A$ are all distinct.

$\text{Partition}(A, l, r)$

1: while true do
2: Choose an element $A[i]$ from $A[l..r]$ uniformly at random
4: $A^+ \leftarrow$ elements in $A[l..r]$ that are larger than $A[i]$.
5: if Size of $A^-$ and size of $A^+$ are both no smaller than a quarter of size of $A[l..r]$ (i.e. $(r - l + 1)/4)$ then
6: Return $A[i]$ (and $A^-$ and $A^+$) as the result of the partition.
7: else
8: end while

Now answer the following questions.

1. What is the worst-case running time of Partition of list of $n$ elements? For a single iteration, what is the probability of statement 6 (the Return) will be called? What is the expected running time for a list with $n$ elements?

The QuickSort algorithm spends most of the time to run Partition subroutine for $A[l, r]$ of different sizes. We now group the array portions as follows: we say a subproblem is of type $i$ if its size is between $n(3/4)^i + 1$ and $n(3/4)^i$.

2. How many type $i$ problems are there? Why? What is the expected running time for Partition to run on a type $i$ subproblem?

3. What is the total expected running time of QuickSort?

2 Counting sort

Do Exercises 8.2-2 (on p. 170). Also, explain why line 9 of the algorithm on p. 168 processes $j$ in a "downto" way?

3 Red-black Tree

Do Exercise 13.3-2. You only need to show the tree after inserting an element.